

## AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

### Listing of Claims:

- 1           1. (Currently amended) A method to facilitate code verification and  
2   garbage collection in a platform-independent virtual machine, comprising:  
3           receiving a code module written in a platform-independent language;  
4           examining the code module to locate a call to a program method within the  
5   code module; and  
6           transforming the code module so that all operands remaining on an  
7   evaluation stack when the program method is called relate to the program method,  
8   wherein transforming the code module involves ensuring that the evaluation stack  
9   includes only elements related to a bytecode that may trigger garbage collection  
10   when the bytecode is executed;  
11           whereby verification and garbage collection of the code module is  
12   simplified.
  
- 1           2. (Original) The method of claim 1, wherein transforming the code  
2   module involves ensuring that local variables hold only values of a single type and  
3   do not hold variables of different types at different times.
  
- 1           3-4 (Canceled).

1           5. (Original) The method of claim 1, wherein transforming the code  
2 module further comprises spilling to memory stack slots that do not include  
3 operands for the call to the program method.

1           6. (Original) The method of claim 5, further comprising filling stack slots  
2 that were previously spilled upon return from the program method.

1           7. (Original) The method of claim 6, wherein the program method is  
2 associated with a single typemap to indicate a type for each variable on the  
3 evaluation stack.

1           8. (Currently amended) An apparatus to facilitate code verification and  
2 garbage collection in a platform-independent virtual machine, comprising:  
3           a receiving mechanism configured to receive a code module written in a  
4 platform-independent language;  
5           an examining mechanism configured to examine the code module to locate  
6 a call to a program method within the code module; and  
7           a transforming mechanism configured to transform the code module so  
8 that all operands remaining on an evaluation stack when the program method is  
9 called relate to the program method, wherein transforming the code module  
10 involves ensuring that the evaluation stack includes only elements related to a  
11 bytecode that may trigger garbage collection when the bytecode is executed;  
12           whereby verification and garbage collection of the code module is  
13 simplified.

1           9. (Original) The apparatus of claim 8, wherein transforming the code  
2 module involves ensuring that local variables hold only values of a single type and  
3 do not hold variables of different types at different times.

1           10-11 (Canceled).

1           12. (Original) The apparatus of claim 8, further comprising a spilling  
2 mechanism configured to spill to memory stack slots that do not include operands  
3 for the call to the program method when transforming the code module.

1           13. (Original) The apparatus of claim 12, further comprising a filling  
2 mechanism configured to fill stack slots that were previously spilled upon return  
3 from the program method.

1           14. (Original) The apparatus of claim 13, wherein the program method is  
2 associated with a single typemap to indicate a type for each variable on the  
3 evaluation stack.

1           15. (Currently amended) A computer system to facilitate code verification  
2 and garbage collection in a platform-independent virtual machine, comprising:  
3           a central processing unit;  
4           a memory system;  
5           a port for communicating with an external client;  
6           a bus to couple the central processing unit, the memory system, and the  
7 port;  
8           a receiving mechanism within the central processing unit configured to  
9 receive a code module written in a platform-independent language;  
10          an examining mechanism configured to examine the code module to locate  
11 a call to a program method within the code module; and  
12          a transforming mechanism configured to transform the code module so  
13 that all operands remaining on an evaluation stack when the program method is  
14 | called relate to the program method, wherein transforming the code module

15 | involves ensuring that the evaluation stack includes only elements related to a  
16 | bytecode that may trigger garbage collection when the bytecode is executed;  
17 |       whereby verification and garbage collection of the code module is  
18 | simplified.

1       16. (Original) The computer system of claim 15, wherein transforming the  
2 code module involves ensuring that local variables hold only values of a single  
3 type and do not hold variables of different types at different times.

1       17-18 (Canceled).

1       19. (Original) The computer system of claim 15, further comprising a  
2 spilling mechanism configured to spill to memory stack slots that do not include  
3 operands for the call to the program method when transforming the code module.

1       20. (Original) The computer system of claim 19, further comprising a  
2 filling mechanism configured to fill stack slots that were previously spilled upon  
3 return from the program method.

1       21. (Original) The computer system of claim 20, wherein the program  
2 method is associated with a single typemap to indicate a type for each variable on  
3 the evaluation stack.